Remarks

The present Response is to the Office Action mailed 03/23/2009, made final. Claims 1-28 are presented for examination.

Claim Objections

4. Claim 18 is objected to because of the following informalities: Claim 18 recites "identification of at least one a plurality". Said claim should be amended to read "identification of at least one of a plurality." (emphasis added) Appropriate correction is required.

Applicant's response:

Applicant herein amends claim 18 as suggested by the Examiner.

Claim Rejections - 35 USC § 101

6. Claims 1-11 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims 1-11 recite a software application executing on a server. A software program per se is non statutory. Thus claims 1-11 are rejected under 35 U.S.C. 101 for failing to fall within a proper statutory category of invention and for failing to be structurally and functionally interconnected with the newly claimed server in such a manner to, in and of itself, enable any usefulness to be realized. The software application must be claimed in combination with an appropriate computer readable medium whereby the server could then access and execute the stored program. Appropriate correction is required.

Applicant's response:

Applicant herein amends claim 18 to specifically recite, "A software application executing from memory on a computerized server providing automated notification of applied structural changes to electronic information pages hosted on a data packet network comprising:" Applicant believes claim 1, as amended, is now statutory as the

software executes from memory of the computerized server thereby creating a new machine.

Claim Rejections - 35 USC § 103

8. Claims 1-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over DaCosta et al (US-6,826,553 11/30/04) in view of Weinberg et al (US-6,360,332 03/19/02) in further view of Heninger (US-6,029,207 02/22/00).

-In regard to substantially similar independent claims 1 and 12, DaCosta teaches an application for enabling automated notification of applied structural changes to electronic information pages on a network comprising:

an interface for enabling users to build and modify network navigation and interaction templates using functional logic blocks for automatically navigating to and interacting with interactive electronic information pages on the network (column 2, lines 11-30; column 5, lines 30-67)(Figs. 1 & 7);

a navigation interface for integrating the software application to a proxy-navigation system for periodic execution of the templates (column 5, lines 19-20: "automatically repeat these steps in a scheduled manner or when requested");

a change notification module for indicating a navigation and interaction routine has failed and for creating a data file associated with the failed routine (column 18, lines 43-67: "it is known the script has failed...and proper notifications sent to individuals or entities responsible for the operation of the failing script by email…for example"; column 19, lines 1-15); and

sending proper notifications of the failed script to the developer upon failure of the script (column 6, lines 9-13 & 35-41; column 18, lines 53-67; column 19, lines 1-15). DaCosta does not specifically teach storing the data file in a data repository with a point-of-failure indication, parameters associated with the failed routine, and an identifier of the associated electronic information page subjected to the navigation. Weinberg teaches storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines (column 6, lines

19-22), and upon failure of the routine, creates a data file (column 2, lines 39-40; column 3, lines 29-43; column 6, lines 19-22; column 17, lines 10- 52)(Fig. 5F), the data file comprising a point-of-failure indication within the failed routine identifying the logic block of the template that failed (Fig. 5F: column 17, lines 17-21), parameters of the failure (column 17, lines 35-43), an identifier of the associated electronic page (columns 17-18: lines 62-12)(Fig. 5F: "URL: www.mercint.com"), and stores the data file in the data repository sending notification of the action to the developer (column 2, lines 39-40; column 6, lines 15-23). It would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process of the failure along with the an identifier of the associated web page, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67).

DaCosta teaches wherein functional logic blocks were part of the navigation and interaction templates containing all of the possible navigation and interaction instructions required by the navigation system-interface module as defined by the a given user/developer (column 2, lines 20-31: "scripts...that locates and extracts data...precisely locating and extracting the select data with a granularity specified by the user" & lines 57-67: "capability for a user to specify…in an automated manor"; column 5, lines 39-55: "learn and store navigation paths...dialogs and forms that need to be filled...login name and password"; column 7, lines 16-28: "captures each user-generated event."; columns 7-8, lines 55-5: "automatically repeatedly query a web site...upon a single exemplematic query"; column 9, lines 5-44). Neither DaCosta nor Weinberg specifically teach wherein the defined functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29). It would have been obvious to one of ordinary

skill in the art at the time of the invention for the functional logic blocks of DaCosta to have been modular parts of the navigation and interaction templates, because Heninger taught that computer software developers realize that modular interacting software components provide the advantages of being more easily designed, generated, tested, installed, and maintained as well leading to better computer products at a minimal cost (column 1, lines 20-67; column 2, lines 1-24). Thus the modular software components of Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script.

-In regard to dependent claims 2, 13, and 19, DaCosta teaches wherein the network could be the Internet (column 2, line 13: "Internet") and wherein the electronic information page was a web page (column 2, line 13: "web site") on the network.

-In regard to dependent claim 3, DaCosta teaches wherein the logic blocks include site logic blocks, automated site-login blocks, and automated site-registration blocks (column 2, lines 55-67; column 5, lines 37-43).

-In regard to dependent claim 4, DaCosta teaches wherein the software application was an Internet based application executing and running on a server (column 18, lines 33-41: "scripts are stored at a central repository that is accessible through the Internet").

-In regard to dependent claim 5, DaCosta teaches wherein the application was accessible through a network browser (column 2, lines 10-30: "Browser").

-In regard to dependent claim 6, DaCosta teaches wherein the templates are test routines executed for determining success or failure of the routine (column 6, lines 9-13 & 35-41; column 18, lines 54-65).

-In regard to dependent claim 7, DaCosta teaches wherein the templates are executable instruction orders containing logic blocks (column 2, lines 55-67).

-In regard to dependent claim 8, DaCosta teaches wherein the functional logic blocks are modular and self-installable within the templates (column 2, lines 55-67)(Fig. 2: 60, 70, 80, 90).

-In regard to dependent claim 9, DaCosta teaches wherein the data files are human readable and are accessed by developers for the purpose of affecting updating of the navigation templates (column 18, lines 54-67).

-In regard to dependent claim 10, DaCosta teaches wherein the developers access the application via individual computerized workstations (column 18, lines 34-67)(Fig. 7: "User Developer").

-In regard to dependent claim 11, DaCosta teaches wherein the error notification and data file are performed in the event failure or a client's personalized navigation template (column 6, lines 9-13 & 35-41; column 18, lines 34-67).

-In regard to dependent claim 14, DaCosta teaches wherein the software application was an Internet (column 2, line 13: "Internet") based application executing and running on a server (column 18, lines 26-40).

-In regard to dependent claims 15 and 16, DaCosta teaches wherein a single server system hosting both the proxy navigation software and the software application (column 18, lines 26-40).

-In regard to dependent claim 17, DaCosta teaches wherein software application and the proxy navigation software are integrated as a single application enabling both

functions of navigation according to navigation templates and notifying and recoding failed instances of navigation (column 18, lines 26-67).

-In regard to independent claim 18, DaCosta teaches a method for receiving automated notification of random structural changes applied to electronic information pages hosted on a network comprising:

-establishing notification of a failed navigation and interaction routine executed for the purpose of navigating to and interacting with an electronic information page (column 6, lines 9-13 & 35-41; column 18, lines 34-67: "email or pager notification").

-creating an instance of the failed routine associated with the cause of failure (column 18, lines 43-67: "it is known the script has failed...and proper notifications sent to individuals or entities responsible for the operation of the failing script by email...for example"; column 19, lines 1-15);

-accessing the notification of the of the failed routine for review purposes (column 6, lines 9-13 & 35-41; column 18, lines 34-67: i.e. developer accesses failed script for re-teaching purposes);

-being able to navigate to the electronic information page identified in the recorded instance (column 6, lines 9-13 & 35-41; column 18, lines 34-67: i.e. developer accesses failed script for re-teaching purposes);

-accessing source information associated with the electronic information page identified in the recorded instance (i.e. re-teaching a new navigation and extraction script by accessing the source information).

-creating new logic block according to the source information and according to information contained in the recorded instance (column 6, lines 9-13 & 35-41; column 18, lines 34-67);

installing the new logic block into existing navigation templates that depend on the updated information for successful function (column 6, lines 9-13 & 35-41; column 18, lines 34-67; column 19, lines 1-15).

DaCosta does not specifically teach wherein the instance of the failed navigation routine was stored for future review including parameters associated with the failed routine that included identification of at least one of a plurality of logic blocks used to build the navigation template. Weinberg teaches storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines (column 6, lines 19-22), and upon failure of the routine, creates a data file (column 2, lines 39-40; column 3, lines 29-43; column 6, lines 19-22; column 17, lines 10-52)(Fig. 5F), the data file comprising a point-of-failure indication within the failed routine and identifying the logic block of the template that failed (Fig. 5F: column 17, lines 17-21), parameters of the failure (column 17, lines 35-43), an identifier of the associated electronic page (columns 17-18: lines 62-12)(Fig. 5F: "URL: www.mercint.com"), and stores the data file in the data repository sending notification of the action to the developer (column 2, lines 39-40; column 6, lines 15-23). It would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process of the failure along with the an identifier of the associated web page, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67).

DaCosta teaches wherein functional logic blocks were part of the navigation and interaction templates containing all of the possible navigation and interaction instructions required by the navigation system-interface module as defined by the a given user/developer (column 2, lines 20-31: "scripts...that locates and extracts data...precisely locating and extracting the select data with a granularity specified by the user" & lines 57-67: "capability for a user to specify...in an automated manor"; column 5, lines 39-55: "learn and store navigation paths...dialogs and forms that need to be filled...login name

and password"; column 7, lines 16-28: "captures each user-generated event."; columns 7-8, lines 55-5: "automatically repeatedly query a web site...upon a single exemplematic query"; column 9, lines 5-44). Neither DaCosta nor Weinberg specifically teach wherein the functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29). It would have been obvious to one of ordinary skill in the art at the time of the invention for the functional logic blocks of DaCosta to have been modular parts of the navigation and interaction templates, because Heninger taught that computer software developers realize that modular interacting software components provide the advantages of being more easily designed, generated, tested, installed, and maintained as well leading to better computer products at a minimal cost (column 1, lines 20-67; column 2, lines 1-24). Thus the modular software components of Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script.

-In regard to dependent claim 20, DaCosta teaches wherein the navigation routine was performed according to a test navigation template (Fig. 2: i.e. according to the navigation and extraction scripts).

-In regard to dependent claim 21, DaCosta teaches wherein the navigation routine was performed according to a client navigation template (Fig. 7: "User").

-In regard to dependent claim 22, DaCosta teaches wherein the recorded instance of the failed routine was created in the form of a data file and stored in a data repository (column 18, lines 54-67).

-In regard to dependent claim 23, DaCosta teaches wherein the recorded instance of the failed navigation routine was accessed by a software developer (column 6, lines 9-13 & 35-41; column 18, lines 54-67).

-In regard to dependent claim 24, DaCosta teaches wherein navigation was performed by the developer utilizing an instance of a browser installed on a computerized workstation (column 2, lines 11-30).

-In regard to dependent claim 25, DaCosta teaches wherein the new logic was in the form of a modular logic block installable to a navigation template (column 6, lines 9-13 & 35-41; column 18, lines 54-67).

-In regard to dependent claim 26, DaCosta teaches wherein the new logic block self-installs to a depended navigation template (column 6, lines 9-13 & 35-41; column 18, lines 42- 67: "ensure each of the users has a corrected script as soon as possible, i.e., as soon as it is downloaded to the central repository...running the script").

-In regard to dependent claim 27, DaCosta teaches testing the new logic before the implementation (column 19, lines 1-15: "determine whether it is operating correctly").

-In regard to dependent claim 28, DaCosta teaches creating more than one logic block within a navigation template and wherein more than one block could fail (column 6, lines 9-13 & 35-41; column 18, lines 34-67; column 19, lines 1-15).

Applicant's response:

Applicant argues that the Examiner is not properly presenting the art when rejecting applicant's claim limitations in the present Office Action. The Examiner provides broad teachings, paragraphs and pages of 09/465,028, without any comment as to what claimed limitation each portion of the art teaches. Applicant points out that the

teaching in the CIP document 09/465,028 drastically differs from teachings in the DaCosta patent. Applicant respectfully requests the Examiner make it clear as to what claimed limitations are taught in 09/465,028. Applicant believes the Examiner places an undue burden on Applicant when presenting a patent in which the priority date is not adequate to qualify for prior art against applicant's claimed invention, while relying on another document, 09/465,028, for the priority date, with no specific comments as to what claim limitations are in 09/465,028. Applicant believes that if the Examiner must rely on 09/465,028 to reject applicant's claimed invention, the burden is on the Examiner to specifically point out what portions of applicant's claim limitations are taught in said document, and what portions are taught in the DaCosta reference.

The Examiner states DaCosta teaches , "enabling users to build and modify network navigation and interaction templates using functional logic blocks for automatically navigating to and interacting with interactive electronic information pages on the network (column 2, lines 11-30; column 5, lines 30-67)(Figs. 1 & 7);" Applicant respectfully disagrees with the above interpretation of DaCosta by the Examiner and every other instance in the present Office Action where the Examiner relies upon DaCosta to teach building and modify network navigation and interaction templates using functional logic blocks.

Applicant argues site-logic blocks comprise software modules providing logic and navigation/interaction templates that include identifiable interaction tasks. DaCosta 09/465,028 teaches storing portions of recordings of users accessing a site with URLs, recording clicks and commands, and then playing them back to achieve Web page access (Fig. 4, i.e. recording script 200). Said recordings or portions of recordings cannot read on applicant's site logic blocks, as claimed. Applicant respectfully requests the Examiner specifically point out in the relied upon reference having the relied upon priority date, 09/465,028, to teach applicant's claimed site logic blocks. The Examiner's failure to do so supports applicant's arguments that teachings of DaCosta, relied upon by the Examiner, are actually not taught or suggested in 09/465,028.

The Examiner states Weinburg teaches, "storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines (column 6, lines 19-22), and upon failure of the routine, creates a data file (column 2, lines 39-40; column 3, lines 29-43; column 6, lines 19-22; column 17, lines 10- 52)(Fig. 5F), the data file comprising a point-of-failure indication within the failed routine identifying the logic block of the template that failed (Fig. 5F: column 17, lines 17-21), parameters of the failure (column 17, lines 35-43), an identifier of the associated electronic page (columns 17-18: lines 62-12)(Fig. 5F: "URL: www.mercint.com"), and stores the data file in the data repository sending notification of the action to the developer (column 2, lines 39-40; column 6, lines 15-23)."

Applicant respectfully disagrees that the indication of failure in Weinburg identifies the failed logic block, as claimed. Applicant argues that Weinberg also provides a facility for recording user interactions. The art of Weinberg actually teaches executing recorded interactions within a stored Web page in order to successfully navigate the page. In applicant's invention, as claimed, modular functional logic blocks are used to build navigation templates, in this manner when navigation fails only replacement or repair of a logic block occurs, sometimes automatically, rather than the requirement of re-recording user interactions as in DaCosta and Weinberg.

The Examiner states, "Neither DaCosta nor Weinberg specifically teach wherein the defined functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29). It would have been obvious to one of ordinary skill in the art at the time of the invention for the functional logic blocks of DaCosta to have been modular parts of the navigation and interaction templates, because Heninger taught that computer software developers realize that modular interacting software components provide the advantages of being more easily designed, generated, tested, installed, and maintained as well leading to better computer products at a minimal cost (column 1, lines 20-67; column 2, lines 1-24). Thus the modular software components of

Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script."

Applicant points out that the Examiner does admit that DaCosta and Weinberg fail to teach wherein the defined functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Applicant points out that DaCosta 09/465,028 and Weinberg also fail to teach defined functional logic blocks in the defined interaction scripts, as claimed.

Applicant points out that the Heniger invention relates to an apparatus and method for the

dynamic linking of computer software components. Specifically, the field of this invention is that of computer systems where the controlling program logic is divisible into at least two components. One of the software components must provide at least one function or data item accessible by a different software component. The Examiner has still failed to provide valid art which teaches upon failure of a test routine, creates the data file, the data file comprising a point-of-failure indication within the failed routine identifying the functional logic block in the associated template.

Therefore, applicant believes independent claims 1, 12 and 18 are patentable over the art provided by the Examiner. Claims 2-11, 13-17 and 19-28 are patentable on their own merits, or at least as depended from a patentable claim.

Specifically, the Examiner states DaCosta teaches, "DaCosta teaches wherein the software application was an Internet based application executing and running on a server (column 18, lines 33-41: "scripts are stored at a central repository that is accessible through the Internet"). Applicant respectfully disagrees. DaCosta 09/465,028 specifically teaches Web Bands application stored on and executing from a user's personal workstation.

The Examiner states, "DaCosta teaches wherein the templates are executable instruction orders containing logic blocks (column 2, lines 55-67)." As previously argued in earnest by applicant, DaCosta fails to teach logic blocks, as claimed. The relied upon

portion of DaCosta is a summary of the new matter of invention and not contained in the valid prior art reference of 09/465,028.

The Examiner states, "DaCosta teaches wherein the functional logic blocks are modular and self-installable within the templates (column 2, lines 55-67)(Fig. 2: 60, 70, 80, 90)." There is absolutely no disclosure in 09/465,028 supporting self-installable modular logic blocks. The Examiner has clearly admitted that DaCosta fails to teach modular site logic blocks. Again, applicant respectfully requests the Examiner point out in DaCosta 09/465,028, where the above limitations of applicant's claims are taught.

Summary

As all of the claims, as argued above, have been shown to be patentable over the art presented by the Examiner, applicant respectfully requests reconsideration and the case be passed quickly to issue.

If any fees are due beyond fees paid with this amendment, authorization is made to deduct those fees from deposit account 50-0534. If any time extension is needed beyond any extension requested with this amendment, such extension is hereby requested.

Respectfully Submitted,
Tim Armandpour et al.

By */Donald R. Boys/*
Donald R. Boys
Reg. No. 35,074

Central Coast Patent Agency, Inc.
3 Hangar Way, Suite D
Watsonville, CA 95076
831-768-1755